

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

THIS PAGE BLANK (USPTO)

(19) 日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平 8 - 2 9 2 8 8 7

(43) 公開日 平成 8 年 (1996) 11 月 5 日

(51) Int. Cl. °

G 0 6 F

9/38

識別記号

3 3 0

庁内整理番号

F I

G 0 6 F

9/38

3 3 0

C

技術表示箇所

審査請求 未請求 請求項の数 6

F D

(全 1 2 頁)

(21) 出願番号 特願平 8 - 58296

(22) 出願日 平成 8 年 (1996) 2 月 21 日

(31) 優先権主張番号 特願平 7 - 62198

(32) 優先日 平 7 (1995) 2 月 24 日

(33) 優先権主張国 日本 (J P)

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目 6 番地

(72) 発明者 島田 健太郎

東京都国分寺市東恋ヶ窪 1 丁目 280 番地 株

式会社日立製作所中央研究所内

(72) 発明者 花輪 誠

東京都国分寺市東恋ヶ窪 1 丁目 280 番地 株

式会社日立製作所中央研究所内

(72) 発明者 山本 一道

東京都国分寺市東恋ヶ窪 1 丁目 280 番地 株

式会社日立製作所中央研究所内

(74) 代理人 弁理士 笹岡 茂 (外 1 名)

最終頁に続く

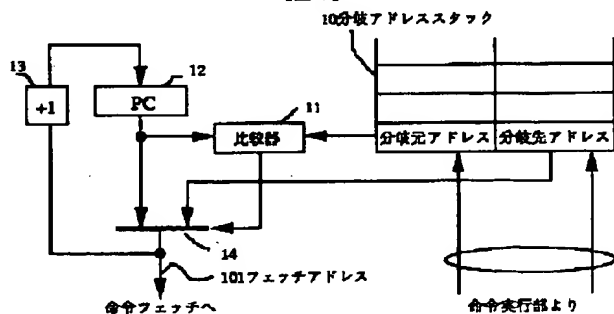
(54) 【発明の名称】 命令の分岐方法およびプロセッサ

(57) 【要約】

【課題】 プログラムのコンパイル時に生成される分岐予約命令を用いて、パイプライン化されたプロセッサでの命令の分岐動作を効率化することにある。

【解決手段】 分岐元アドレスと分岐先アドレスの複数の組を保持する分岐アドレススタック（以下、スタック）10と、現在の命令フェッチアドレスを保持するプログラムカウンタ（PC）12と、スタック10の保持する最新の組の内の分岐元アドレスとPC12の値とを比較する比較器11と、比較の結果が一致のときに命令フェッチアドレスをPC12の値からスタック10の保持する最新の組の内の分岐先アドレスに切り替えるセレクタ14を備えたプロセッサで、命令列中の分岐点に先立って、分岐予約命令によりスタック10に分岐先アドレスと分岐元アドレスの組を予約しておき、命令の読み込みが分岐点に達したならば速やかに次の命令読み込みアドレスを分岐先に切り替えることで、分岐動作を行う。

【図 1】



【特許請求の範囲】

【請求項 1】 プロセッサにおける命令の分岐方法であつて、
分岐予約命令の実行に応じて該分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスを予約するステップと、
命令の読み込みを行うアドレスが前記予約した分岐元のアドレスに達したか否か比較判定するステップと、
前記比較判定の結果、前記命令の読み込みを行うアドレスが前記予約した分岐元のアドレスに達したとき、該命令の読み込みを行うアドレスを前記予約した分岐先のアドレスに切り替えるステップからなることを特徴とする命令の分岐方法。

【請求項 2】 請求項 1 記載の命令の分岐方法において、
前記分岐予約命令が複数個順次実行されたとき、前記分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスを予約するステップを複数回繰り返すことにより複数の分岐を予約し、
前記比較判定するステップにおいては、前記予約された複数の分岐の内最も最近に予約された分岐元のアドレスを選択して前記比較判定をすることを特徴とする命令の分岐方法。

【請求項 3】 請求項 1 または請求項 2 記載の命令の分岐方法において、
分岐予約命令の実行により分岐元のアドレスと分岐先のアドレスの予約が行われた後で分岐が行なわれる前記分岐元のアドレスに達する前に前記分岐元のアドレスと分岐先のアドレスの予約を条件的に取り消す命令が配置されている場合、該条件的に取り消す命令の実行に応じて前記予約した分岐元のアドレスと分岐先のアドレスを条件的に取り消すステップを設けることにより命令の条件付き分岐を行なうことを特徴とする命令の分岐方法。

【請求項 4】 命令読み込み手段と、少なくとも一つの命令解読手段と、少なくとも一つの命令実行手段を備え、前記命令読み込み手段と命令解読手段と命令実行手段が並列に動作可能なプロセッサにおいて、
前記命令読み込み手段は、命令の読み込みを行うアドレスを保持するプログラム・カウンタと、
分岐予約命令が実行されたとき該分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスの組を格納保持する分岐アドレス組保持手段と、
前記プログラム・カウンタの保持する値と前記分岐アドレス組保持手段の保持する分岐アドレス組の内の分岐元のアドレスとを比較する比較手段と、
前記比較手段による比較の結果が一致を示すとき、前記プログラム・カウンタの保持する値を前記分岐アドレス組保持手段の保持する分岐アドレス組の内の分岐先アドレスの値に切り替えて出力する切り替え手段を備え、
命令の読み込みを行うアドレスが分岐予約命令の指定し

た分岐元のアドレスと一致したとき、分岐予約命令が指定した分岐先のアドレスの命令へ分岐を行うことを特徴とするプロセッサ。

【請求項 5】 請求項 4 記載のプロセッサにおいて、
前記分岐アドレス組保持手段は、分岐先アドレスと分岐元アドレスの組を複数保持でき、
前記比較手段は、前記分岐アドレス組保持手段に保持された分岐アドレス組の内の最新の組を選択し、該選択した組の分岐元のアドレスの値と前記プログラム・カウンタの保持する値とを比較することを特徴とするプロセッサ。

【請求項 6】 請求項 4 または請求項 5 記載のプロセッサにおいて、
前記分岐アドレス組保持手段は、前記命令実行手段からの取り消し信号を受けて該取り消し信号の指定する分岐元のアドレスと分岐先のアドレスの組を消去する手段を備えることを特徴とするプロセッサ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、プロセッサに関し、特に命令の読み出しと命令の実行がパイプライン化されて並列に行われるプロセッサに適する命令の分岐方法とそのための装置に関するものである。

【0002】

【従来の技術】まず、従来のパイプライン化されたプロセッサにおける分岐動作について述べる。なお、ここで述べる従来のパイプライン化されたプロセッサにおける分岐動作については、文献「Computer Architecture: A Quantative Approach」(David A. Patterson and John L. Hennessy, Morgan Kaufmann Publishers, Inc., 1990)で、特に6章においてよくまとめられている。一般的なパイプライン化されたプロセッサでは、高速化のために、命令の実行とこれに後続する命令の読み込みが並列に行われている。しかし分岐命令が実行されてかつ分岐が生じる時には、分岐命令に後続する命令は実行されないで、その読み込みは無駄になる。更に従来では、分岐命令の実行の結果初めて分岐先が定まるので、分岐先の命令の読み込みが行えるようになるまでには時間がかかる。パイプライン化されたプロセッサでは、このように分岐命令においてパイプラインによる命令の読み込みと実行の並列動作が不可となるので、大きく性能を落とす要因となっていた。

【0003】そこで従来では、分岐命令に続く後続命令を1命令だけそのまま実行することによって、後続命令の読み込みを有効に利用しようとする方策が採られていた。これは、見かけ上分岐が生じるのが分岐命令の次の命令の後に遅れるように見えることから、遅延分岐方式と呼ばれる。プロセッサの回路方式を工夫すれば、更に

分岐先アドレスの計算も後続命令を読み込んでいる間に行うことが可能であり、そのように構成すれば、後続命令の次に続けて分岐先の命令の読み込みを行うことも出来る。このようにすることによってパイプライン化されたプロセッサにおいても分岐動作を効率良く行おうとしていた。

【0004】ところが、オペレーティング・システムなどプロセスの管理を行うようなプログラムでは、分岐先アドレスの計算をレジスタ等の値を用いて動的に行う分岐命令が用いられている。しかし、このような複雑な分岐先アドレスの計算を行うと、後続命令をただかだか1命令読み込んでいる時間では計算が終了しないということも生じる。また分岐命令につづいて実行される後続命令1命令は、特にその分岐命令が条件分岐命令の時は分岐の生起／不生起に関わらず実行できることが条件となる。しかし、そのような命令を常に見付けることは、プログラムの性質にもより、完全にはできないことが多い。

【0005】さらに、近年では高性能化のために、一つのプロセッサ内に複数の命令実行パイプラインを用意し、複数の命令を同時に読み込んで並列に実行させる、スーパースカラ方式やVLIW (Very Long Instruction Word) 方式と呼ばれる命令単位の並列実行方式を採用するプロセッサも多くなって来ている。このようなプロセッサでは、分岐命令とそれに後続する命令がしばしば同時に読み込まれてしまうので、遅延分岐方式のように後続命令の読み込み時間で分岐命令の実行時間を隠すことができない。結果として、分岐命令（を含む複数命令）の読み込みから次に実行すべき分岐先の命令を読み込めるようになるまでの時間が顕著に現われてしまい、このような命令単位の並列実行方式を採用するプロセッサにおける大きな問題となっていた。

【0006】このような問題に対する従来の解決策は、分岐予測方式である。これには、静的な分岐予測と動的な分岐予測の二つの基本方式がある。静的な分岐予測では、分岐命令のタイプと分岐先が前方か後方かなどの静的な情報によって分岐が起こるかどうかを予測する。動的な分岐予測では、その分岐命令で過去に分岐したかどうかの履歴をある程度記録しておき、その履歴に基づいて一定のアルゴリズムで次に分岐が起こるかどうかを予測する。プロセッサ内の命令の読み込み部では、読み込んだ（複数）命令中に分岐命令があるかどうかを検出する。分岐命令があった場合には、分岐予測の結果に従って次に読み込む命令を後続の命令にするか分岐先の命令にするかを決定する。この命令読み込み部における分岐命令の検出は、すぐ次の命令読み込みに間に合うように充分素早く行う必要がある。従来の分岐予測方式では、分岐命令があったアドレスをその分岐先アドレスと組にして記録しておく、内容（の一部）での検索が可能な連

想メモリを用いたテーブルを用意するという方法を探っている。即ち、それぞれの分岐命令の初回の実行でその分岐命令が置かれていたアドレスをその実行の結果得られる分岐先アドレスと共に記録しておき、新しく命令を読む込む度にそのアドレスが記録されていないかどうかそのテーブルを検索する。このようにすることによって、それぞれの分岐命令が2回目以降、早期に検出でき、分岐先のアドレスも得ることができる。またテーブルには分岐命令の置かれているアドレス及び分岐先のアドレスと組にして、その分岐命令の過去に分岐発生履歴に関する情報を記録しておけば、分岐予測に関する機構も同時に実装することが可能である。

【0007】このように従来では分岐予測方式によって、パイプライン化されたプロセッサで分岐をさらに効率化しようとしていた。しかし、従来では分岐命令の初回の実行によってその分岐命令の置かれているアドレスと分岐先のアドレスを登録するので、初回の実行は常に予測ができない。また登録するテーブルも限られた大きさであり、多数の分岐命令が存在するときはLRU (Least Recently Used) など一定のアルゴリズムで入れ替えを行わなければならない。入れ替えの結果追い出された分岐命令は、再び予測不可となる。またさらに大きな問題としては、分岐予測の精度がある。即ち、分岐予測方式では、実際の分岐命令の実行に先だってあくまで予測をするだけであるので、予測がはずれることも充分有り得る。このような予測の精度は、先に述べたような分岐予測の方法にもよるが、実行するプログラムの性質にもよる。即ち、実行する度に毎回分岐の生起／不生起が変化するような分岐命令を含むようなプログラムでは、予測の精度を上げることは非常に困難である。予測がはずれた場合には、予測の結果に基づいて読み込んでいた命令を一旦すべて取り消して、元の分岐点まで戻って命令の読み込みからやり直さなければならない。その結果、予測しなかった場合よりもかえって余分な時間がかかる場合も多い。

【0008】

【発明が解決しようとする課題】本発明は、分岐先の決定は多くの場合、プログラムのコンパイル時に予め静的に、あるいはサブルーチンからの戻り（リターン）などのように動的に決定される場合にも、命令列中で分岐をしなければならない分岐点に充分先立って決定できることに着目し、前述した問題点を解消しようとするものである。本発明の目的は、プログラムのコンパイル時に生成される分岐予約命令によって、パイプライン化されたプロセッサで命令の読み込みと命令の実行が並列に行われていても、効率良く分岐を行う方法およびプロセッサを提供することにある。

【0009】

【課題を解決するための手段】上記目的を達成するため、本発明は、プロセッサにおける命令の分岐方法であ

り、分岐予約命令の実行に応じて該分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスを予約するステップと、命令の読み込みを行うアドレスが前記予約した分岐元のアドレスに達したか否か比較判定するステップと、前記比較判定の結果、前記命令の読み込みを行うアドレスが前記予約した分岐元のアドレスに達したとき、該命令の読み込みを行うアドレスを前記予約した分岐先のアドレスに切り替えるステップからなるようにしている。さらに、前記分岐予約命令が複数個順次実行されたとき、前記分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスを予約するステップを複数回繰り返すことにより複数の分岐を予約し、前記比較判定するステップにおいては、前記予約された複数の分岐の内最も最近に予約された分岐元のアドレスを選択して前記比較判定をするようにしている。さらに、分岐予約命令の実行により分岐元のアドレスと分岐先のアドレスの予約が行われた後で分岐が行なわれる前記分岐元のアドレスに達する前に前記分岐の予約を条件的に取り消す命令が配置されている場合、該分岐予約命令を条件的に取り消す命令の読み出しに応じて前記予約した分岐元のアドレスと分岐先のアドレスを条件的に取り消すステップを設けることにより命令の条件付き分岐を行なうようにしている。また、命令読み込み手段と、少なくとも一つの命令解読手段と、少なくとも一つの命令実行手段を備え、前記命令読み込み手段と命令解読手段と命令実行手段が並列に動作可能なプロセッサにおいて、前記命令読み込み手段は、命令の読み込みを行うアドレスを保持するプログラム・カウンタと、分岐予約命令が実行されたとき該分岐予約命令が指定する分岐元のアドレスと分岐先のアドレスの組を格納保持する分岐アドレス組保持手段と、前記プログラム・カウンタの保持する値と前記分岐アドレス組保持手段の保持する分岐アドレス組の内の分岐元のアドレスとを比較する比較手段と、前記比較手段による比較の結果が一致を示すとき、前記プログラム・カウンタの保持する値を前記分岐アドレス組保持手段の保持する分岐アドレス組の内の分岐先アドレスの値に切り替えて出力する切り替え手段を備え、命令の読み込みを行うアドレスが分岐予約命令の指定した分岐元のアドレスと一致したとき、分岐予約命令が指定した分岐先のアドレスの命令へ分岐を行うようにしている。さらに、前記分岐アドレス組保持手段は、分岐先アドレスと分岐元アドレスの組を複数保持でき、前記比較手段は、前記分岐アドレス組保持手段に保持された分岐アドレス組の内の最新の組を選択し、該選択した組の分岐元のアドレスの値と前記プログラム・カウンタの保持する値とを比較するようにしている。さらに、前記分岐アドレス組保持手段は、前記命令実行手段からの取り消し信号を受けて該取り消し信号の指定する分岐元のアドレスと分岐先のアドレスの組を消去する手段を備えるようにしている。

【0010】

【実施例】本発明の実施例を図によって説明する。図1は、本発明の一実施例を図示したものである。本発明はプロセッサの、特に命令フェッチ（読み込み）アドレス生成部において実施されるので、図1では命令フェッチアドレス生成部に限って示してある。プロセッサ全体の構成については、一例を図2に掲げておく。図1では、命令実行部で分岐予約命令が実行されることにより、分岐元アドレスと分岐先アドレスが分岐アドレススタック10に登録される。上記分岐予約命令はプログラムのコンパイル時に予め作成され、分岐点に充分先立って命令列中に配置される。すなわち、分岐点における分岐先の決定は、多くの場合、予め「静的」＝「プログラムのコンパイル時」に或いはサブルーチンからの戻り（リターン）などのように動的に決定される場合にも、命令列中で分岐をしなければならない分岐点に充分先立って行なうことができる。そして、コンパイル時に分岐元アドレスと分岐先アドレスを指定する分岐予約命令が分岐点に充分先立って命令列中に配置される。命令フェッチへは通常はプログラムカウンタ（PC）12より命令読み込みアドレスがフェッチアドレス101のパスへ出力されている。また、一回の命令読み込みごとに次の命令読み込みアドレスがインクリメンタ（“+1”加算器）13より計算されて、プログラムカウンタ12に書き戻される。プログラムカウンタ12の出力は、また比較器11にも入力され、分岐アドレススタック10に登録されている分岐元アドレスと一致していないかどうか、比較されている。図1の例では、分岐アドレススタック10には複数の分岐元アドレスと分岐先アドレスの組が登録可能であり、その中で最新の組が選ばれて、その分岐元アドレスが比較器11に入力されている。比較器11で一致が検出されると、分岐アドレススタック10に登録されたその組の分岐先アドレスが取り出されセクタ14によりフェッチアドレス101のパスへ出力されて、命令フェッチに用いられる。該分岐先アドレスが分岐アドレススタック10から取り出されると、この分岐先アドレスと分岐元アドレスの組が分岐アドレススタック10から消去され、上の段に格納されていた組が最新の組となる。また同時に、分岐先アドレスがインクリメンタ13を通してプログラムカウンタ12に書き戻されるので、分岐先アドレスから以後再び連続して命令を読み込むことが可能となる。

【0011】図2では、プロセッサ1と主記憶2について示してある。プロセッサ1では、図1のような構成を取る命令フェッチアドレス生成部21を持つほかに、読み込んだ命令を解読する命令デコード部22、解読結果に従って命令を実行する命令実行部23、及びプロセッサ内でデータを蓄えるレジスタファイル24を備えている。また主記憶2は、プロセッサ1の外の記憶要素であり、プロセッサ1とは、フェッチアドレス101、フェ

ッチ命令202、データアドレス203、書き込みデータ204、読み出しデータ205の各データパスによって接続されている。主記憶2は、まず、フェッチアドレス101のパスに出力された命令アドレスに従って記憶されている命令をフェッチ命令202のパスに返す。またデータアドレス203に出力されたデータアドレスに従って書き込みデータ204のパスに出力されたデータを記録したり、読み出しデータ205のパスに記憶されたデータを出力したりする。この主記憶2にはキャッシュを含んでも良い。主記憶2よりフェッチ命令202のパスに読み出された命令は、フェッチアドレス101による命令アドレスと共に命令デコード部22に送られる。命令デコード部22では、命令を解読して解読結果を命令実行部23に送ると同時にレジスタファイル24より必要なレジスタの内容を読み出して、同じく命令実行部23に送る。命令実行部23では、送られた命令解読結果及びレジスタの内容を用いて命令を実行し、実行結果をレジスタファイル24に書き戻したり、主記憶25に対してデータの読み出し/書き込みを行ったりする。特に分岐予約命令の時には、実行の結果定まる分岐元アドレスと分岐先アドレスの値をそれぞれ分岐元アドレス206、分岐先アドレス207のパスに出力し、命令フェッチアドレス生成部21に送る。図2では、以上のような命令フェッチアドレス生成部21、命令デコード部22、及び命令実行部23の動作は並列に行われているものとする。即ち、これによって命令の処理がパイプライン化されている。

【0012】図3は、命令フェッチアドレス生成部21についての、本発明の別の実施例である。図3では、分岐アドレスレジスタ30に一組だけの分岐元アドレス及び分岐先アドレスの値が保持されている。また命令実行部からは、図1の実施例では分岐元アドレスと分岐先アドレスの設定信号だけであったが、図3の例ではそれに加えて設定取り消し信号も備えられている。分岐先アドレスが分岐アドレスレジスタ30から取り出されると、この分岐先アドレスの組、すなわち分岐先アドレスと分岐元アドレス、が分岐アドレスレジスタ30から消去されることは図1の場合と同様である。命令実行部において条件取り消し命令が実行され、その条件取り消し命令で指定された条件（例えば、フラグA=1）が成立していると、この設定取り消し信号が出力され、これにより分岐アドレスレジスタ30に一旦登録されていた分岐元アドレスと分岐先アドレスが取り消されて無効となる。これにより条件分岐の処理を実現している。その他の比較器11、プログラムカウンタ12、インクリメント13、セクタ14の構成は図1と同じである。

【0013】図8は、さらに別の実施例であり、図1に示す構成に図3に示す設定取り消し信号を備えたものである。この設定取り消し信号により分岐アドレススタック10の最下段の組、すなわち最新の予約の組、を取り

消すようにしている。なお、分岐アドレススタック10に設定された複数の予約の組の内どれを取り消すかを指定するようにしてもよい。

【0014】図4は、分岐予約命令の例および分岐の予約を取り消す命令の例である。図4(a)は最も基本的な分岐予約命令の形式であり、分岐元アドレス(f r o m _ a d d r)、及び分岐先アドレス(t o _ a d d r)の二つを命令中で値を指定する即値オペランドとして持つ。この二つの即値オペランドの内のどちらか、あるいは両方とも、現在のプログラムカウンタの値からの相対値としても良い。相対値によるアドレス計算が必要であれば、例えば図2の例では命令実行部23で計算させることができる。

【0015】図4(b)は他の分岐予約命令の形式であり、図4(a)に対し、分岐先アドレスをレジスタ(r e g)の内容で指定できるようにした例である。この命令が実行されると、上記レジスタ(r e g)の内容を参照して、分岐先アドレスを求め、命令で指定された分岐元アドレスと求められた分岐先アドレスを組として分岐アドレススタックあるいは分岐アドレスレジスタに設定する。このような命令形式は、サブルーチンからの戻り(リターン)など、実行時に動的に分岐先が決まる場合や、図4(a)では二つの即値に対して、(コンパイラにより機械語を生成する際の)命令のエンコードに必要なビット数が不足して充分大きな値を設定できない場合にも有効である。

【0016】図4(c)は、一旦予約した分岐を条件的に取り消す命令の例である。これを図4(a)または

(b)のような分岐予約命令と組み合わせれば、条件分岐の処理を実現できる。対象とするプロセッサで図8の実施例のように同時に複数の分岐予約ができる場合には、どの予約を取り消すのかも指定する必要があるが、ここでは図8の説明で述べたように、常に最新の予約の組を取り消すと定める。そのため図4(c)では、取り消すための条件のみを命令中のc o n d i t i o nで指定する形式となっている。すなわち、条件として、例えば、フラグA=1を指定すれば、フラグA=1が成立している場合、設定取り消し信号が分岐アドレススタック10に与えられ、最新の予約の組が強制的に取り消される。このように複数の組から最新のものを選ぶ機能は、図1のような実施例では、比較器11に供給する分岐元アドレスをどの組から選ぶか決定されている際に用いられるものであるので、これを取り消す組の選択にも利用することが可能である。

【0017】図4(d)は、一旦予約した分岐を条件的に取り消す命令の別の例である。これは、指定された二つのレジスタ% a及び% bの内容を比較し、更にc o n d i t i o nで指定した条件が成立していれば、図4(c)と同じく最も最近に予約された分岐を取り消す。この命令は、条件分岐の処理によく現われる比較を行う

命令と、図4(c)にあるような取り消す条件のみを指定する条件付き取り消し命令との融合命令である。このような命令を設けることにより、特に図4(c)のような予約した分岐を取り消すための命令を別個に実行する必要がなくなり、命令数が削減される。また図4(d)では、命令で指定しなければならないオペランドは比較の対象となるレジスタ二つと、分岐を取り消すための条件の計三つであり、簡単であるので、命令のエンコードにも難点はない。もし、命令のエンコードに難点が予想されるような複雑な条件を指定するなど、図4(d)のように一命令で実現するには複雑な実施を行う場合には、図4(c)のような条件のみを指定する分岐取り消し命令と、通常の比較命令などを組み合わせて行えばよい。

【0018】次に本発明を、従来の分岐方式と比較してみる。図5は、本発明による分岐予約を行うプロセッサと、従来のプロセッサでの条件分岐を行う命令列の例である。まず、図5(a)にC言語による条件処理の例を掲げる。図中指定された条件($a == b$)が成立するとthen部が、不成立の時にはelse部が実行される。図5(a)のような条件分岐の処理を従来の分岐命令を持つプロセッサの命令列で実現した例が図5(b)である。図5(b)では、変数a及びbの値が予めレジスタ%a及び%bにそれぞれ保持されているものとする。まず比較命令 `compare %a, %b` により変数a、bの値が等しいかどうか調べられ、等しくない時、条件分岐命令 `branch not_equal, L1` によりelse部の処理を行うラベルL1以下へ分岐する。変数a、bの値が等しい時はラベルL1への分岐は行われず、then部の処理を行うラベルL0以下が実行される。そしてthen部の最後に、ラベルL1以下のelse部の処理を行わずに、処理をラベルL2以下の処理に続けるために、無条件分岐命令 `branch L2` によりラベルL2へ分岐する。

【0019】これに対し、本発明による分岐予約方式の命令列は図5(c)のようになる。図5(c)では、初めに条件が不成立だったとき実行されるラベルL1以下のelse部への分岐を命令 `branch_target L0, L1` により予約しておく。図5(c)でも図5(b)と同じく、変数a及びbの値は予めレジスタ%a及び%bにそれぞれ保持されているものとする。これにより、比較付き条件取り消し命令 `compare_and_pop %a, %b, equal` でまず変数a、bの値の比較を行う。そして変数a、bの値が等しいという条件が成立していた時のみ、else部への分岐の予約を取り消す。実際には、この予約の取り消しも分岐点L0の命令の読み込み前に終了できるようにコンパイラで命令列のスケジューリングの最適化が行われるものとする。もし最適化が行われず条件取り消し命令による予約の取り消しが分岐点の命令の実行の直前

となり、その読み込みに間に合わなかったときには、予約に従って読み込まれていた命令を取り消すなどの制御が必要である。逆に図5(b)のような従来のプロセッサの命令列との大きな違いは、分岐点L0までの命令がすべて予約に関する命令であり、分岐点L0(の直前)には何の命令も置く必要はないので、分岐点L0よりかなり以前に遡る命令のスケジューリングも可能であると言えることである。即ちコンパイラにおける命令スケジューリングの最適化に関する自由度が大きい。図5(c)において、条件取り消し命令によって分岐予約が取り消された時にはラベルL0以下のthen部が実行されるが、その先頭で命令 `branch_target L1, L2` でthen部の終り(else部の直前)L1からの分岐を予約し、then部の処理の終了後、ラベルL1以下のelse部の処理をすることなく、滞り無く次のラベルL2以下の処理を続けられるようにしている。この場合も分岐点となるラベルL1(の直前)には何の命令も置く必要もなく、命令のスケジューリングに大きな自由度がある。

【0020】図6は、本発明による分岐予約を行うプロセッサでの、サブルーチン呼び出し/戻りを行う命令列の例である。サブルーチン呼び出しでは、サブルーチンの先頭アドレスは多くの場合静的に決定でき、本発明の分岐予約方式を容易に適用できる。またサブルーチンからの戻り(リターン)は、戻り先が一般には動的に決定されるが、その決定の時点はサブルーチンの呼び出し時であり、実際の分岐点であるサブルーチンの終りに充分先立って決定されている。これらのことから、サブルーチンの呼び出し/戻りにも、本発明による分岐予約方式を適用して効率を高めることができる。

【0021】図6において、サブルーチンL2の呼び出しを命令 `branch_target L1, L2` によって実際の呼び出し点L1に充分先立って予約している。またサブルーチンからの戻りアドレスを設定するために、呼び出し点L1までに命令 `mov L1, %r1` で戻りアドレスL1(図6の例では呼び出し点に同じ)をレジスタ%r1に設定しておく。呼び出された側のサブルーチンL2では、サブルーチンの終りL3までにレジスタ%r1の内容を分岐先とする分岐予約命令 `branch_on_register L3, %r1` により戻り先を予約しておく。以上のような命令列によって、呼び出し点L1よりサブルーチンL2が呼び出され、サブルーチンの終りL3から再び戻り点L1へ戻るという動作が分岐予約により実現される。

【0022】次に実際のパイプラインの動作として、本発明による分岐予約と従来の例を比較してみる。図7は、図1、図3あるいは図8で示したような本発明による分岐予約を行うプロセッサで実際の分岐が生じている時のパイプライン動作と、従来のプロセッサの分岐のパイプライン動作の比較である。ここで、Fは命令フェッ

チ、Dは命令デコード、Eは命令実行、Mはメモリ・アクセス、Wはレジスタ・ライトバック、また、矢印は分岐アドレスの供給を示す。

【0023】図7(a)は従来の分岐方式による分岐処理時のパイプライン動作の例である。分岐命令の直前

(分岐点)の命令、分岐命令、分岐命令の結果次に分岐先の命令を読み込めるまでのディレイスロット1サイクルと、分岐先の命令の各実行サイクルをパイプラインに沿って図示してある。図7(a)のように、従来の方式では、最低限分岐命令が読み込まれて命令デコードサイ

クル(D)まで達しないと分岐先が決定できない。このため図7(a)では、分岐の直前(分岐点)の命令から分岐先の命令までの間に分岐命令とそのディレイスロットの2サイクルが存在する。これに対し、図7(b)に本発明による分岐予約方式のパイプライン動作の例を示す。図7(b)では、分岐点には、分岐のためには何ら命令を必要としない。分岐点の命令(分岐の直前の命令)のフェッチサイクル(F)において、命令の読み込みが分岐点まで達したことが検出されると、図1、図3、図8で説明したような機構で速やかに分岐先の命令

の読み込みを開始できる。分岐点の命令と分岐先の命令の間にも余計なサイクルは不要である。このようにパイプラインにおける動作を非常に効率化できる。

【0024】図9A~Cは、本発明の予約による分岐のパイプラインの動作と、従来の予測による分岐のパイプライン動作の違いを示した例である。図9A~Cにおいて、Fは命令のフェッチ(読み込み)、Dは命令のデコード(解読)、Eは命令の実行、Mはメモリ(キャッシュまたは主記憶)のアクセス、Wは命令の結果のレジスタへの書き戻しの動作を行うパイプライン・ステージを示す。また、分岐先アドレスの供給を実線の矢印で表している。更に、分岐点の命令フェッチの開始時点(SP1)、分岐先の命令フェッチの開始時点(SP2)として示してある。

【0025】図9Aは、従来の予測による分岐の方法において、分岐が生じたときに予測が誤っていた場合のパイプライン動作の例である。図9Aにおいて、命令1は分岐点(分岐命令の直前)の命令、命令2は分岐命令、命令3は分岐命令に後続する、誤った予測によってフェッチされた命令である。また命令4は正しい分岐先の命令である。分岐の予測の誤りは分岐命令(命令2)がDステージで解読されて、検出可能となる。このとき、分岐命令に後続する命令3は既にFステージで読み込まれているので、命令3のデコードおよびそれ以降の動作を取り消さなければならない。また同時に正しい分岐先の命令4をフェッチし直す必要がある。正しい分岐先、即ち命令4のアドレスは、分岐命令のDステージ以降で生成されるので、結果として、分岐先の命令フェッチ(SP2)は、分岐点の命令フェッチ(SP1)から3サイ

クルかかることになる。

【0026】図9Bは、従来の予測による分岐の方法において、分岐が生じたときに予測が正しかった場合である。図9Bにおいて、命令1は分岐点(分岐命令の直前)の命令、命令2は分岐命令、命令3は予測によってフェッチされた分岐先の命令である。分岐の予測は分岐命令(命令2)のフェッチによって行われるので、図のように、分岐命令の読み込みに続いて、遅滞なくすぐ次のサイクルで分岐先の命令3のフェッチを開始することが可能である。結果として分岐点の命令フェッチ(SP1)から2サイクルで分岐先の命令フェッチ(SP2)が始まっている。

【0027】図9Bのように正しい予測が行われるためには、分岐命令2が1回以上実行されて、分岐先が記録されている必要がある。また同時に予測に用いているアルゴリズムが該分岐命令の動作を正しく予測できなければならない。従って、分岐命令2の初回の実行、および予測のアルゴリズムが正しく予測できないときには、パイプラインの動作は図9Aに示したようになり、3サイクルかかることになる。

【0028】これらに対し、図9Cに本発明の予約による分岐のパイプライン動作を示す。図9Cにおいて、命令1は分岐予約命令、命令2は通常の命令、命令3は分岐点の命令、命令4は分岐先の命令である。パイプラインの動作では、まず、分岐予約命令(命令1)の実行によりEステージで、分岐点のアドレスと分岐先のアドレスの両方が登録されて、分岐の予約が行われる。分岐予約命令は、命令のフェッチが分岐が生じる分岐点に達する充分前に行われるように、コンパイラによってプログラムのコード生成時に配置される。その後、図9Cでは、一個以上の通常命令(命令2)がフェッチされ、実行される。そして、命令のフェッチが分岐点の命令3に達したことがFステージで検出されると、命令のフェッチアドレスがすぐに分岐予約命令1で登録された分岐先アドレスに切り替えられ、遅滞なく分岐先の命令4のフェッチが行われる。このとき、分岐が生じる分岐点においては、分岐に関する命令はなんら実行される必要はない。分岐点の命令3も命令2と同じく、任意の通常命令でよい。結果として、分岐点の命令3のフェッチ(SP1)から1サイクルで、分岐先の命令4のフェッチが行われている。図9Cのようなパイプライン動作は、本発明の予約による分岐では、分岐予約命令の初回の実行から常に可能である。

【0029】図10A~Bは、本発明による分岐予約命令を用いてプログラムをコンパイルしてコード生成を行う例を示したものである。図10Aでは、プログラムをコンパイルする際の、コンパイラの全体の操作の例を示してある。まずステップ110でコンパイルするプログラムのソースが読み込まれ、ステップ120で構文解析及び文法チェックが行われる。次にステップ130ではプログラムの意味解析が行われ、ステップ140で制御

フローグラフを、ステップ150でデータフローグラフが生成される。これらのフローグラフの情報を元にステップ160で大域的最適化及び局所的最適化が行われ、ステップ170で実際の機械コードが生成されて、出力される。

【0030】図10Bは、図10Aのステップ170における機械コード生成を詳細に示したものである。図10Bにおいて、まずステップ171で各基本ブロック毎にその中に含まれる演算操作が決定される。次にステップ172でそれぞれの演算操作に対応する機械命令を生成する。ステップ173で機械命令コードにおける最適化を行う。そしてステップ174で分岐予約命令を各基本ブロックの先頭に挿入する。最後にステップ175で生成した機械コードの列を出力する。

【0031】図10Cは実際にコンパイルされるプログラムのソースの構造の例を示している。また図10Dは図10Cのプログラムをコンパイルした結果を示している。これらにおいて、基本ブロックはA~Dの4個が示されている。図10Cでは、プログラムの構造を基本ブロックA~Dの接続として示してある。即ちプログラムは基本ブロックAから開始され、基本ブロックAの終わりから基本ブロックBまたはCへ条件分岐を行う。基本ブロックB及びCは基本ブロックDへ無条件に分岐する。基本ブロックDをすべて実行するとプログラムが終了する。

【0032】このような分岐を本発明の分岐予約命令を用いて実現すると、図10Dに示したような機械コード列となる。即ち、基本ブロックAの先頭には、Aの終わりからCの先頭への分岐を予約する分岐予約命令が置かれる。その後、基本ブロックAの演算を行う機械命令と共に、条件分岐を行うための条件生成の命令が置かれ、これを受けて分岐予約の取り消しを行う条件取消命令も置かれる。基本ブロックAの命令列の直後には基本ブロックBの命令列が配置され、基本ブロックAの命令列において生成された条件が成立していると、条件取消命令により、基本ブロックAの先頭で予約された分岐が取り消され、命令の実行はそのまま基本ブロックBに移る。このとき、条件取消命令で指定する条件は分岐を取り消す条件、即ち分岐しない条件である。この条件が不成立の時には、条件取消命令が実行されても予約の取り消しは行われず、基本ブロックAの命令列の最後から（最後の命令をフェッチすると）、予約された分岐が実行され、命令の実行が基本ブロックCに移る。基本ブロックBの先頭には、Bの終わりからDの先頭への分岐を予約する分岐予約命令が置かれ、Bの終わりからは（無条件に）Dの先頭への分岐が起こる。従って、図10Dのように基本ブロックCの命令列を基本ブロックBの命令列の直後に配置することが可能であり、基本ブロックDの命令列をCの直後に配置しておけば、図10Cに示した基本ブロックB及びCから基本ブロックDへの無条件分

岐が表現できる。

【0033】図11A~Eは、本発明の予約による分岐について、更に詳細な動作の例を説明したものである。図11Aにおいて、メモリ内にはアドレス99に、分岐点アドレスが102、分岐先アドレスが1000の分岐予約命令が置かれている。プロセッサ内において、図11Aでは、このアドレス99の分岐予約命令の読み込みは終了しており、命令実行部において実行が開始されている。命令実行部における実行の結果、分岐アドレススタックには、分岐点アドレス102と分岐先アドレス1000のアドレスペアが登録され、分岐が予約される。その後、図11Bにおいてプログラムカウンタ（PC）により保持されている命令フェッチアドレスが分岐アドレススタックに登録された分岐点アドレス102と等しくなると、図11Cにおいて命令フェッチアドレスが登録された分岐先1000に切り替えられる。図11Cで命令フェッチアドレスの切り替えが行われている間に分岐点の命令102の読み込みが完了し、図11Dで分岐先1000の命令の読み込みが行われている間に並列に実行されている。同様に、分岐先命令の読み込みは図11Dで完了し、図11Eで後続のアドレス1001の命令が読み込まれている間に、これと並列に実行される。

【0034】図12A~Cは、従来の予測による条件分岐のパイプライン動作の様子と、本発明の分岐予約による条件分岐のパイプライン動作の样子の違いの例を示したものである。図12A~Cにおいて、Fは命令のフェッチ（読み込み）、Dは命令のデコード（解読）、Eは命令の実行、Mはメモリ（キャッシュまたは主記憶）のアクセス、Wは命令の結果のレジスタへの書き戻しの動作を行うパイプライン・ステージを示す。また、図12A、Bでは、分岐先アドレスの供給を実線の矢印で、分岐点の命令フェッチの開始時点をSP1、分岐先の命令フェッチの開始時点をSP2として示してある。図12Cでは、条件分岐に関する条件取消命令のフェッチ開始時点をSP1とし、分岐後の命令フェッチの開始時点をSP2としている。

【0035】図12Aは、従来の予測による条件分岐で、予測が誤っていた場合の例である。図12Aで、命令1は比較命令、命令2は条件分岐命令、命令3は誤った予測によってフェッチされた命令、命令4は正しい分岐先の命令である。命令3のDステージ以降の処理は、条件分岐命令（命令2）がDステージで解読されて予測が誤っていたことが検出されると、取り消される。その際、正しい分岐先のアドレスが生成されて、Fステージに伝達され、命令4のフェッチが行われる。この結果、分岐点、即ち条件分岐命令（命令2）の直前の比較命令1のフェッチ時点SP1から、正しい分岐先の命令4のフェッチ時点SP2まで3サイクルかかる。

【0036】図12Bは、従来の予測による条件分岐で、予測が正しかった場合のパイプライン動作の例であ

る。図12Bで、命令1は比較命令、命令2は、条件分岐命令、命令3は予測による（この場合は正しい）分岐先の命令である。図のように、条件分岐命令（命令2）のフェッチで分岐先のアドレスが予測されるので、予測が正しい場合には、すぐ次のサイクルで分岐先の命令4のフェッチを行うことができる。結果として、分岐点の命令1のフェッチSP1から分岐先の命令4のフェッチまで2サイクルになる。このように分岐先のアドレスまで正しく予測できるためには、図9Bとの場合と同じく、条件分岐命令2が一度以上実行されて、分岐先のアドレス計算が行われ、何等かの分岐予測の手段により記録されていなければならない。また同時に予測に用いるアルゴリズムが条件分岐の成立／不成立を正しく予測できなければならない。

【0037】図12Cは、本発明の分岐予約による条件分岐のパイプライン動作の例を示したものである。特に、分岐が生じる条件が成立せず、分岐しなかった場合を示す。図12Cで、命令1は分岐予約命令、命令2は条件取消命令、命令3は通常の命令、命令4は分岐点にあたる通常の命令、命令5は分岐点に連続する（分岐しなかったときの）命令である。これらの内、命令3、4は任意の有効な通常命令を充てることが出来る。図12Cでは、分岐予約命令（命令1）の実行によりEステージで分岐点と分岐先のアドレスが予約される。分岐先アドレスは、図の点線の矢印で示すように、分岐点に後続する命令5のFステージに伝達されようとする。しかし条件取消命令（命令2）の実行によりEステージで取り消しが生じて（図中の“X”印）、分岐が生じず、命令5は分岐点の命令4に後続する命令となる。以上の動作により、図12Cでは、分岐点の命令4のフェッチから遅滞なく1サイクルで後続する命令5のフェッチを行うことが出来る。条件分岐に関する条件取消命令（命令2）のフェッチ時点（SP1）から数えても、3サイクルで条件分岐を行うことが出来る。

【0038】このように、本発明の予約により分岐する方法においても、条件分岐を効率良く行うことができる。

【0039】

【発明の効果】本発明により、命令単位の並列実行を行うプロセッサを含むパイプライン化されたプロセッサにおいて、命令の実行と命令の読み込みの並列動作をなんら妨げることなく、効率的に分岐動作を行わせることができる。

【図面の簡単な説明】

【図1】本発明によるプロセッサの命令フェッチアドレス生成部の一実施例を示す図である。

【図2】プロセッサの全体構成の例を示す図である。

【図3】本発明によるプロセッサの命令フェッチアドレス生成部の第2の実施例を示す図である。

【図4】本発明による分岐予約を行うための分岐予約命

令の例および分岐予約の取り消し命令を示す図である。

【図5】本発明による条件分岐処理と従来技術による条件分岐処理の比較説明のための命令列の例を示す図である。

【図6】本発明による分岐予約命令を用いたサブルーチン呼び出し及び戻りの例を示す図である。

【図7】本発明による分岐予約命令と従来の分岐命令のパイプライン動作の比較を説明するための図である。

【図8】本発明によるプロセッサの命令フェッチアドレス生成部の第3の実施例を示す図である。

【図9A】従来の予測において、分岐が生じたときに予測が誤っていた場合のパイプライン動作の例を示す図である。

【図9B】従来の予測において、分岐が生じたときに予測が正しかった場合のパイプライン動作の例を示す図である。

【図9C】本発明の予約による分岐のパイプライン動作の例を示す図である。

【図10A】プログラムをコンパイルする際の、コンパイラの全体の操作の例を示す図である。

【図10B】図10Aのステップ170における機械コード生成を詳細に示す図である。

【図10C】実際にコンパイルされるプログラムのソースの構造の例を示す図である。

【図10D】図10Cのプログラムをコンパイルした結果を示す図である。

【図11A】本発明の予約による分岐動作の例を説明するための図である。

【図11B】図11Aの分岐動作の続きを説明するための図である。

【図11C】図11Bの分岐動作の続きを説明するための図である。

【図11D】図11Cの分岐動作の続きを説明するための図である。

【図11E】図11Dの分岐動作の続きを説明するための図である。

【図12A】従来の予測による条件分岐で、予測が誤っていた場合のパイプライン動作の例を示す図である。

【図12B】従来の予測による条件分岐で、予測が正しかった場合のパイプライン動作の例を示す図である。

【図12C】本発明の分岐予約による条件分岐のパイプライン動作の例を示す図である。

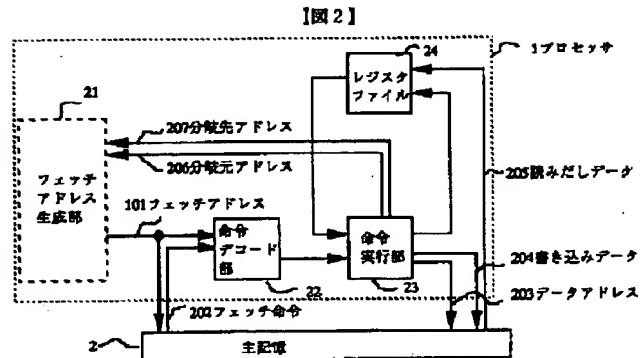
【符号の説明】

- 1 プロセッサ
- 2 主記憶
- 10 分岐アドレススタック
- 11 比較器
- 12 プログラムカウンタ
- 13 インクリメンタ
- 14 セレクタ

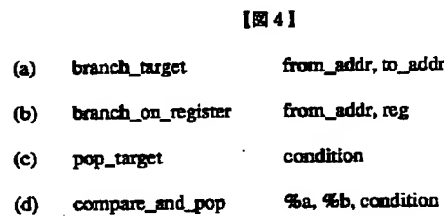
18

- 24 レジスタファイル
30 分岐アドレスレジスタ
101 フェッチアドレス

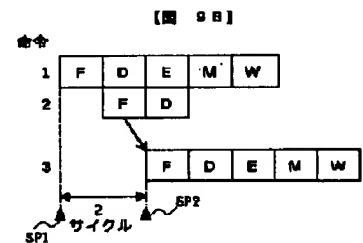
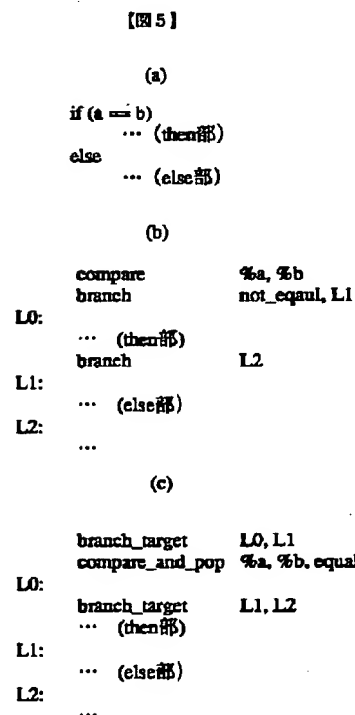
【図2】



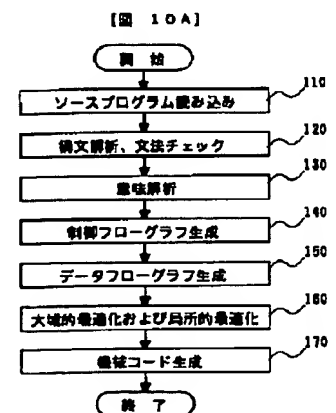
【图 4】



【図 9 B】



【図 10 A】

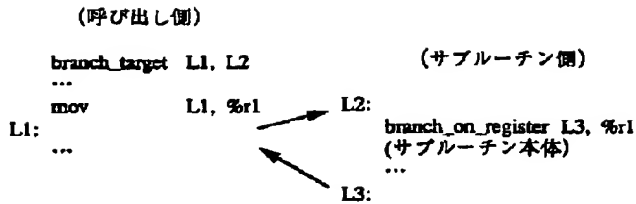


【図6】

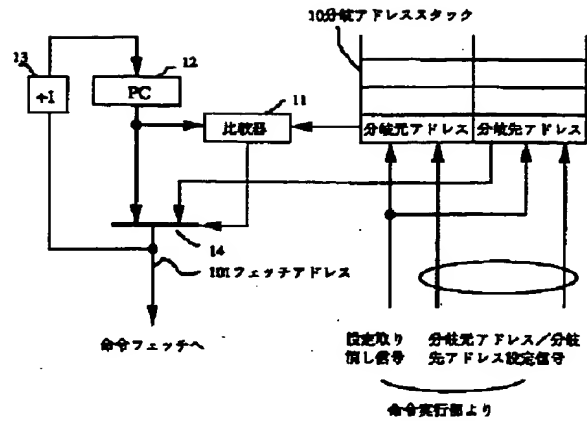
【図8】

【図6】

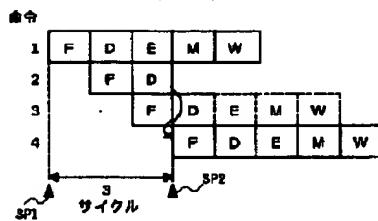
【図8】



【図9A】



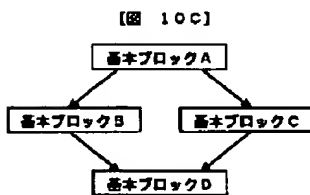
【図9A】



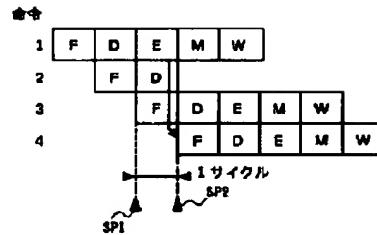
【図9C】

【図10B】

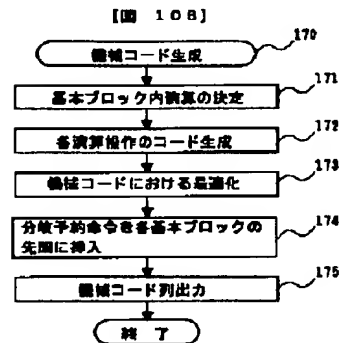
【図10C】



【図9C】

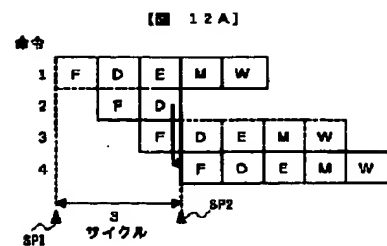


【図10D】



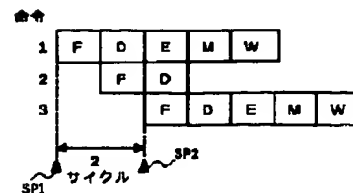
【図10D】

【図12A】

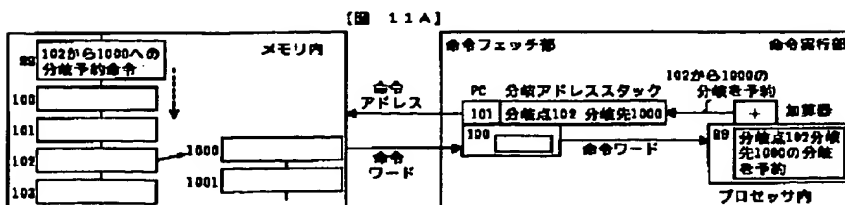


【図12B】

【図12B】

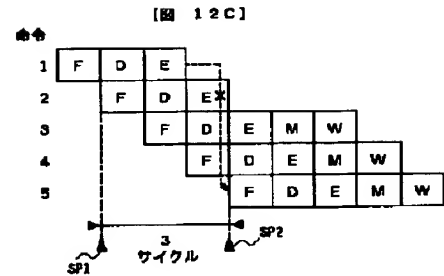
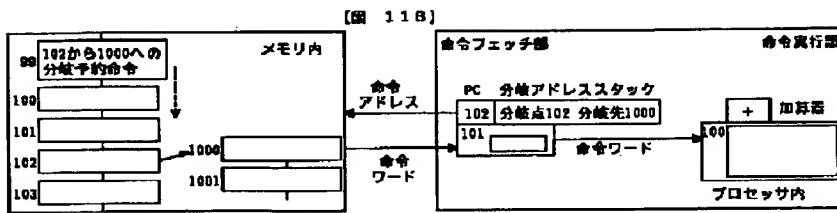


【図11A】

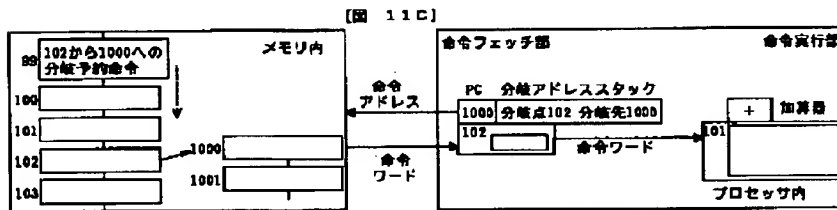


【図 11 B】

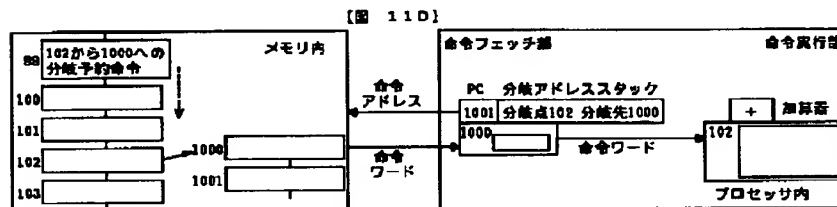
【図 12 C】



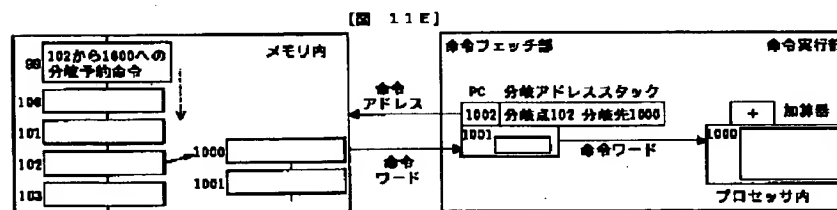
【図 11 C】



【図 11 D】



【図 11 E】



フロントページの続き

(72) 発明者 金子 憲二
東京都国分寺市東恋ヶ窪 1 丁目 280 番地
株式会社日立製作所中央研究所内